IRIDIA

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle

Finite State Automata Synthesis in Boolean Network Robotics

L. Garattoni, C. Pinciroli, A. Roli, M. Amaducci, and M. Birattari

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2012-017

September 2012

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2012-017

The information provided is the sole responsibility of the authors and does not necessarily reflect the opinion of the members of IRIDIA. The authors take full responsibility for any copyright breaches that may result from publication of this paper in the IRIDIA – Technical Report Series. IRIDIA is not responsible for any use that might be made of data appearing in this publication.

Finite State Automata Synthesis in Boolean Network Robotics

Lorenzo Garattoni[*], Carlo Pinciroli, and Mauro Birattari

IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium

Andrea Roli and Matteo Amaducci

DEIS-Cesena, *Alma Mater Studiorum* Università di Bologna, Italy

Abstract

We show that the automatic design of robot control systems based on Boolean networks can be exploited as a method of synthesis of Finite State Automata (FSA). To demonstrate our insight, we study a set of Boolean networks designed to let a robot perform two simple tasks. The networks are configured through a suitable optimization algorithm. We focus our analysis on the structure of the network state space. In the best performing networks, we observed two patterns. First, the network dynamics exploit only a very limited region of the state space. Second, such region is structured in a set of clusters of states. Further analysis revealed that it is possible to derive a compact FSA representation of the network state space in which the states of the FSA correspond to the clusters. We conclude the paper outlining possible perspectives of the proposed approach.

1 Introduction

Boolean Networks (BNs) are a model of Genetic Regulatory Networks (GRNs) [11]. BNs are extremely interesting from an engineering perspective because of their ability to produce complex behaviors, despite the compactness of their description and the simplicity of their implementation. BN dynamics can be studied through traditional dynamical system methods [3, 21]. The use of concepts such as state space, trajectories and attractors, combined with the discrete nature of BNs, enables non-trivial analysis of the behavior of BNs.

In [19], we have shown that BNs can be utilized to control robots. In this paper we continue this line of research, showing that a BN-based controller can be expressed as a Finite State Automaton (FSA).

Analogously to [19], we employ an optimization algorithm to configure the parameters of the BN-controller for a given task. We then analyze the behavior of the best performing BN-controllers. The analysis reveals that such behavior

[*]Corresponding author. Email: lgaratto@ulb.ac.be

is organized into modules, and that it is possible to represent the behavior in the form of a FSA. In other words, in this paper we show an automatic method to derive a FSA-based robot behavior through a BN-based controller.

The automatic design of FSA is not a new concept. In fact, the automated design of compact high-level representations of control systems for intelligent agents is indeed been a challenge in artificial intelligence. Several ways of modeling the behavior of artificial agents exist, but the finite state automaton representation is the oldest and is broadly used.

Among the approaches to obtain a FSA representation of a behavior automatically, Evolutionary Programming is one of the most notable [5, 6]. EP is a paradigm used for the generation of programs, code, algorithms and structures in general, by means of variation and selection mechanisms inspired by natural evolution. Even though EP was shown to produce interesting results in many important applications, several issues are still open about its employment [14]. One of the main issues to address is the choice of the most appropriate representation for the programs to be evolved. In fact, the most suitable representation and the appropriate encoding of the programs into individuals in the evolution process are critical aspects for the performance of EP [16]. We suggest in this work that the automatic configuration of BNs can be exploited as a novel approach, alternative to EP, for the automatic synthesis of FSA. This approach does not suffer from the issue of representation of the programs during the design process. In fact, the FSA is an indirect product of the automatic design of BNs. The optimization algorithm acts only on the BN structure, avoiding the problem of defining the appropriate representation of the behaviors.

The remainder of the paper is structured as follows. In Section 2, we introduce Boolean networks and their employment in robotics. The methodology used to carry out our studies is illustrated in Section 3. In Section 4 we discuss the analysis performed and the results, outlining the heuristics used to obtain the FSA representation of the controllers and the possible perspective of the work. Conclusions and an outlook to future work are given in Section 5.

2 Boolean network robotics

In the remainder of this section we first introduce the BNs and then we describe how they are employed and configured to let robots perform the desired tasks.

2.1 Boolean networks

A Boolean network is a discrete-state and discrete-time dynamical system. Its structure is defined by an oriented graph with N nodes each associated to a Boolean value x_i, $i = 1, ..., N$, and a Boolean function $f_i(x_{i_1}, ..., x_{i_{K_i}})$, where K_i is the number of inputs of node i. The arguments of function f_i are the Boolean values of the nodes whose outgoing arcs are connected to i. The state of the system at time t, with $t \in \mathbb{N}$, is defined as the set of the N Boolean values at t. The state space size is 2^N. Several update schemes can be defined [10], but the most studied is characterized by synchronous and deterministic operations.

BN dynamics can be studied by means of the usual dynamical system methods [3, 21], hence the usage of concepts such as state space, trajectories, attractors and basins of attraction. Recently, the attention of the scientific commu-

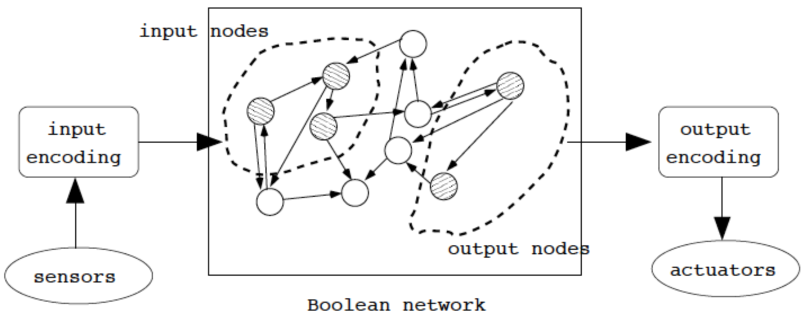


Figure 1: The coupling between BN and robot

nity has focused on the employment of efficient mathematical and experimental methods for analyzing network dynamics and thus have insight into the behavior of a BN system [8, 18, 20]. A special ensemble of BNs that has received particular attention is that of Random Boolean Networks (RBNs), which can capture relevant phenomena in genetic and cellular mechanisms and complex systems in general.

2.2 BN-Robot coupling

To design a BN-based robot control system, we first need to couple the BN to the robot so as to let the BN dynamics guide the robot behavior. For this purpose, some nodes of the network are given special roles. More precisely, we define a set of input nodes and a set of output nodes. This choice characterizes our approach with respect to most of the work performed about the BNs, in which they are considered as isolated systems, even though some notable exceptions exist [2, 4, 12, 15]. The Boolean values of the input nodes are not determined by the network dynamics, but they are imposed according to the robot sensor readings. Similarly, the values of the network's output are used to encode the signals for maneuvering the robot's actuators. Several ways to define the mapping between sensor readings and network's input, and between network's output and actuators are possible. However, the simplest is to define the mapping via a direct encoding. Figure 1 shows the coupling between BN and robot.

2.3 Automatic design methodology

Once a mapping between the BN and the robot is defined, the BN must be designed in order to control the robot's behavior. Our approach consists in treating BN design as a search problem. In fact, the design of a BN that satisfies given criteria can be modeled as a constrained combinatorial optimization problem by properly defining the set of decision variables, constraints and the objective function. The methodology is depicted in Figure 2.

The search algorithm manipulates the decision variables which encode structure and Boolean functions of a BN. A complete assignment of these variables defines an instance of a BN. Then, we couple this network to the robot through

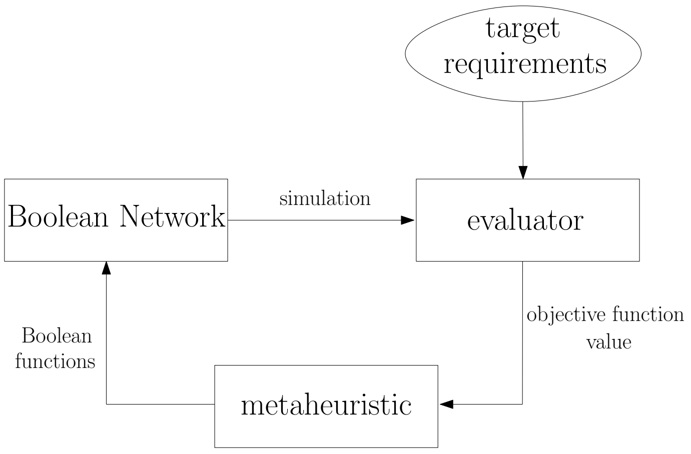


Figure 2: BN design by metaheuristics

the input-output mapping, and subsequently we execute the network. The evaluation of the network at each iteration of the search process is performed in a batch of simulated experiments. The performance of the robot in each experiment is assessed according to a user-defined fitness function, which associates the robot behavior to a numeric evaluation. Finally, the search algorithm exploits this value of performance to proceed with the design process. In particular, the algorithm changes the configuration of the decision variables so as to find networks with better performance evaluations.

3 Methodology

Our previous work [19] showed the effectiveness of BNs in the design of robot control systems through the methodology described in Section 2.3. Here we focus on the properties of the results, in particular on the internal organization of the resulting networks. By analyzing the state space of the best performing networks, in fact, we reveal patterns that characterize the results of the design process.

Since the objective of the first work was to prove the feasibility of the design of BN-robot control systems, we carried out a simple test case: phototaxis and antiphototaxis. The test case ensured the two fundamental features of *dynamic nature* and *memory*. The robot, in fact, must be able to perform both phototaxis and antiphototaxis, and switch between the two behaviors whenever a specific event occurs. The dynamic nature of the task stems from the fact that the time in which this event will occur is unpredictable. In the test case, the unpredictable event consisted of a clap triggered at a random instant during the experiment. To perform the task, the robot needs to keep memory of the perception of the clap in order to select the right behavior to be performed at any instant of time.

In this paper, we turn our attention to the study of the final results. To this aim, it is interesting to analyze how the dynamic nature of the task and the memory influence the organization of the state space in the best performing

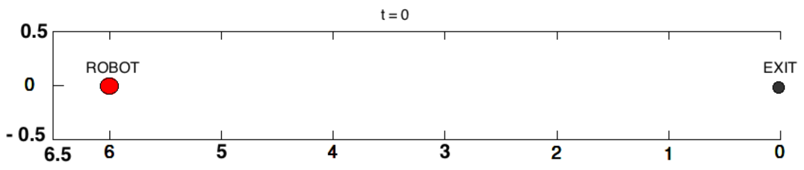


Figure 3: Corridor navigation environment

networks. To better highlight the properties of the networks in each situation, we analyze the two features separately, in two different test cases. Each test case is devised to emphasize one of the two aspects in a more complex scenario than the one of phototaxis and antiphototaxis. This methodology allows us to have deeper insight into the dynamical behavior of BNs depending on the nature of the task to be performed.

3.1 Robot and Simulator

For both experiments, the robots are trained in simulation. The simulation framework we employed is the open source simulator ARGoS [17]. ARGoS is a discrete time, physics-based simulation environment that allows for real-time simulation of large swarms of robots.

The robot simulated in the experiments is the *e-puck* [13]. The e-puck is a small wheeled robot, designed for research and educational purposes. It has a cylindrical body of 7 cm of diameter, equipped with a variety of sensors. For our test cases, we use the 8 infra-red proximity sensors placed along the circular perimeter of the robot and the 3 infra-red sensors pointed directly at the ground in front of the robot. The 3 latter sensors can be used to detect the color of the ground, in greyscale. The actuators utilized, besides the motors of the two wheels, are the 8 red LEDs.

3.2 Corridor navigation

The first test case is designed to explore the features of networks able to perform a *dynamic task*. It consists of a robot that must navigate along a corridor avoiding any collision with the walls and finally reach the exit.

Environment: it consists of a straight corridor of 6.5 m in length and 1 m in width.

Task: in the beginning, the robot is placed within the corridor 6 m far from the exit. During the experiment, the robot must advance along the corridor, avoiding collisions and finally, within the given total execution time $T = 120$ s, reach the exit. See Figure 3 for a representation of the environment in the beginning of the experiment.

During the execution, if a collision between the robot and the walls of the corridor occurs, the experiment is immediately stopped.

Performance measure: the performance assigned to the robot is simply its final distance from the exit (normalized). The smaller is this distance, the better is the robot performance.

BN-robot setup: for successful navigation, the robot needs the 8 proximity sensors to detect the walls and avoid them. At each time step, the readings of the 8 sensors are encoded into the values of the BN input nodes. We use 4 input nodes to encode the readings of the proximity sensors. Thus, the 8 proximity readings are gatherer in pairs. If at least one of the two sensors of the pair exceeds a chosen threshold, the corresponding input node value is set to 1. The pairs are formed in such a way that the robot can detect walls in the four directions north-east, south-east, south-west and north-west.

Once the readings of the sensors are encoded in the input nodes, we perform the network state update and finally we read, decode and utilize the values of the output nodes to set the actuators. Two output nodes are used to set the wheel speeds either to zero or to a predefined, constant value.

For this test case, we set the network size to 20 nodes. We leave for future investigation the analysis on how this value affects performance. In particular, how this value should scale as a function of the task complexity in order to provide a good trade-off between computation cost in simulating the network and size of the network state space.

BN design: the initial topology of the networks, i.e. the connections among the nodes, is randomly generated with $K = 3$ (i.e. each node has 3 incoming arcs) and no self-connections, and it is kept constant during the training. The initial Boolean functions are generated by setting the 0/1 values in the f_i uniformly at random. Our local search strategy works only on the Boolean functions. In particular, at each iteration, the search algorithm changes the configuration of the network by flipping one bit of the Boolean functions. The flip is performed by changing a random entry in the f_i of i, where i is a randomly chosen node, and accepted if the corresponding BN-robot system has a performance not worse than the current one. The evaluation of each network is performed on a set of initial conditions, that form the training set. For this test case, the training set is composed by six different initial orientations of the robot. The six angles are chosen so as to have six equally spaced orientations in the range between $\frac{\pi}{3}$ and $-\frac{\pi}{3}$ (with 0 that is the straight direction of the robot towards the exit). In this manner, the robot must be able to cope with a wide range of different situations and avoid the walls it detects in any direction. The final evaluation assigned to the robot is computed as the average of the performance across the 6 trials. We executed 100 independent experiments, each corresponding to a different initial network. For each experiment the local search was run for 1000 iterations.

3.3 Sequence recognition

The second test case aims to explore the *memory* aspect. The task is sequence recognition [23]. In particular, the robot must learn to recognize a sequence of colors by performing certain actions. This kind of task is more complex than the previous one, because the robot needs a form of memory to be able to choose the next action depending on the past.

Environment: it consists of a straight corridor of 7 m in length and 1 m in width. Along the corridor, the ground is painted to form a striped pattern with three different colors: white (W) represents the background, while black (B) and gray (G) are the symbols of a sequence to be recognized.

Task: in the beginning of the experiment, the robot is placed within the corridor

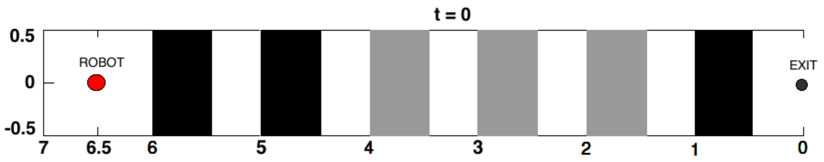


Figure 4: Sequence recognition corridor environment

6.5 m far from the exit. During the experiment, the robot must move along the corridor and reach the exit. Every time the robot encounters a black or gray area in the right sequence, it must turn its LEDs on. Conversely, when the robots encounters the background color or other colors in the wrong order, it must keep its LEDs off. The sequence to be recognized is a cyclic repetition of black followed by gray. By performing the right sequence of actions while moving along the corridor, the robot must be able to reach the exit within the given total execution time, fixed in T=130 s. Figure 4 represents an example of the environment in the beginning of the experiment.

In the environment depicted in Figure 4, the robot must perform the following sequence of actions to achieve the goal (omitting the background color (W) whose corresponding LED correct status is always OFF):

```
Colors along the corridor:   B    B    G    G    G    B
Robot's LEDs status:          ON   OFF  ON   OFF  OFF  ON
```

If the robot, at any instant of time during the execution, performs the wrong action, the experiment is immediately stopped.

Performance measure: The performance assigned to the robot is the final distance from the exit of the corridor (normalized between 0 and 6.5). The value must obviously be minimized.

BN-robot setup: for this task, the robot needs the ground sensor to detect the color of the ground. For our simple application we use only the central sensor. Since we encode three values (W, B, G), at each time step, the reading of the sensor is encoded into the values of two BN input nodes. Four nodes are used to encode the proximity sensors that, even though not strictly needed for the task, can be still useful for the navigation along the corridor.

After the network's state update, the values of the output nodes are decoded and used to set the actuators. Besides the two nodes used to control the wheel speeds, an additional output node is utilized to set the state of the LEDs either to ON or OFF.

For this test case we increased the network size to 30 nodes.

BN design: initial topology and Boolean functions are randomly generated with $K = 3$. The local search strategy, stochastic descent, works only on the Boolean functions leaving the topology unchanged. The evaluation of each network is performed on a set of initial conditions. More precisely, the training set is composed by 10 different randomly generated sequences of colors on the ground. Differently from the corridor navigation case, the robot starts always pointing towards the exit. In this way, the navigation task is simplified so as to focus the complexity on the sequence recognition. The final evaluation of a robot is the average value of the performance across the 10 trials. Due to the high computational cost required by each experiment, we executed only 30 independent experiments with 30 different initial networks for 100000 iterations of the local search.

4 Results and perspectives

The analysis of the results obtained in both experiments allowed us to discover two patterns. First, the dynamics of the networks designed through the methodology described in Section 2.3 utilize a very limited region of the whole potential state space. This means that, somehow, the local search algorithm moves towards networks whose dynamics are more compact. This relationship between the design process and the dynamical features of the networks is notable: the search algorithm works directly only on the network structures, searching for a good behavior while ignoring the dynamics property of the networks. Nevertheless, the analysis shows that the algorithm shapes the BNs so as to have their dynamics confined and compact.

The second pattern observed is the organization of the state space traversed by the final networks in a set of clusters of states, each devoted to perform a specific series of actions. Further analysis about this insight revealed that a compact view of the dynamical behavior of the networks can be provided in terms of a finite state automaton representation.

The compression of the state space
Once the design process has been completed, the focus of the analysis is on the dynamical features of the resulting networks. The first pattern observed is the compression of the dynamics in a confined region of the state space. In order to carry out this analysis, we collected a large number of trajectories, corresponding to different initial conditions, for each BN obtained. Then, we counted the number of different states that each network traversed across all the trajectories and we reported the resulting values in a boxplot. Figure 5 shows the distribution of these values for the corridor navigation test case.

The boxplot shows that the final network dynamics traverse a limited region of the state space. In fact, the 100 final networks, among which about 90 are able to perform the task, use on average around 150 states. This is a very tiny fraction of the whole potential space, whose dimension is 2^N (2^{20} in this case). The same trend emerges from the results of the sequence recognition test case. The networks able to perform the task, 5 out of 100, confirm the same property: the state space usage is on average 200 states out of 2^{30} potential states.

These results provide evidence that the search algorithm indirectly compresses the dynamics while acting on the network structure. Thus, there exists a relationship between the trend of the search process and the dynamical features of the networks, even though search space and state space are not explicitly linked.

The relationship between search process and network dynamics
The trend of the number of states visited along the search process reveals the presence of several cycles of exploration and exploitation phases. This relationship between the search process and the number of states visited has been studied by gathering the trajectories of the networks at each iteration of the

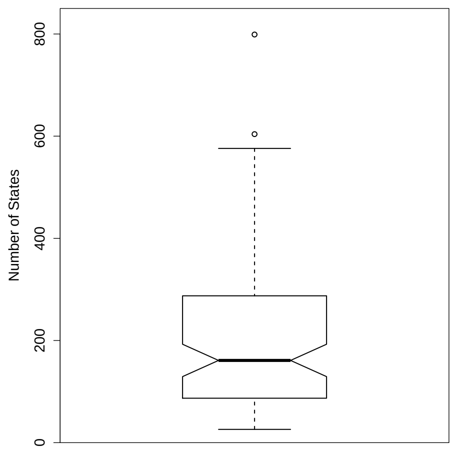


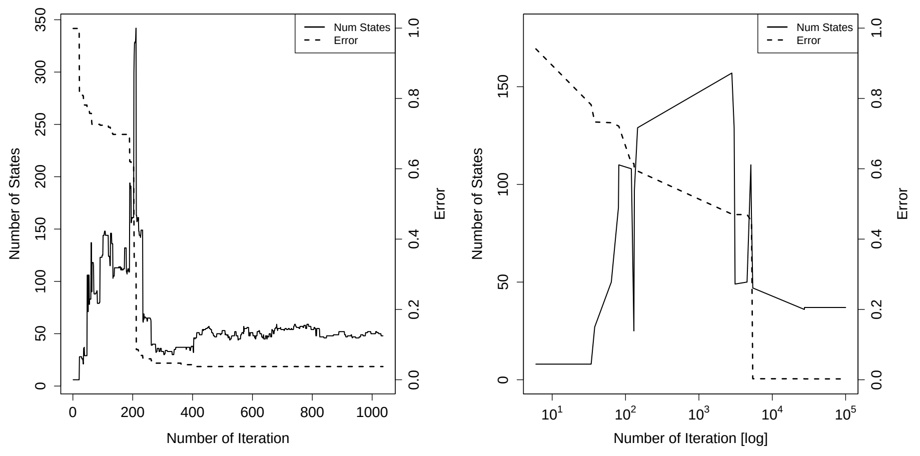
Figure 5: Number of visited states in final networks. Corridor navigation test case

local search, and then counting the number of states traversed. Figure 6 shows the plots of the overlapped trends obtained in two typical cases for both test cases.

The plots show the relationship between the two quantities. In the corridor navigation test case (see Figure 6(a)), the initial number of states is very low. At the beginning, the search algorithm quickly increases this number in order to expand the space of the possible dynamics and, consequently, enhance the probability to find a dynamical behavior able to perform the task. This phase can be seen as an exploration that the design process performs in the state space of the networks. Thus, the point at which the number of states reaches its peak corresponds to the greatest drop of the error function. Once a good solution has been found, a second phase begins. The search algorithm optimizes the solution by compressing its dynamics. The number of states decreases, leading to slight improvements of the error function. We can see this phase as exploitation of the solution found during the exploration phase. Finally, both trends stabilize. The final number of visited states, as discussed, is very limited (≈ 50) with respect to the whole state space.

The exploitation phase has the effect of generalizing the task performed. The compression of the dynamics, in fact, leads the networks to reuse the same sets of states to perform the same sets of actions. The generalization organizes the dynamics of the network so as to have that each region of the state space is devoted to the execution of a specific set of actions. Every time that the robot needs to perform a particular action, the network utilizes the same responsible set of states. In this way, the task is generalized, from a single to a repeated execution. Moreover, this mechanism of generalization optimizes the behavior of the robot. The reuse of the same regions of space and the compactness of the dynamics reduce the presence of transient states that might negatively affect the responsiveness, and consequently the performance, of the robot. The slight improvements of the error functions achieved during the generalization phase support this observation.

The plot about the sequence recognition test case (Figure 6(b)) confirms the



(a) Corridor navigation (b) Sequence recognition

Figure 6: States number and error function trends. Corridor navigation (a), sequence recognition (b).

observations drawn, with some relevant differences. The higher complexity of the task, which lies in the memory requirement, forces the local search algorithm to execute both the exploration and the exploitation phases several times. In each phase, the algorithm tries to improve the solution as much as it can until it reaches a plateau. With the exploration phase it searches for new solutions by expanding the portion of state space utilized. As soon as this strategy does not produce any new improving solution, the algorithm starts to compress and generalize the dynamical behavior. Another notable thing is that, differently from the first case, the greatest drop of the error function has been obtained during the generalization phase. This stems from the nature of the task: the main difficulty, in fact, lies in the generalization of the sequence to recognize, from a single to a cyclic recurrence of any length.

After further analysis of the phenomena, future works could exploit these features to improve the performance of the design methodology. For instance, the local search algorithm could be guided explicitly in the alternation of phases, speeding up the search and avoiding it to get stuck in plateau.

From BNs to FSA

Compactness and organization in the state space of the resulting BNs make it possible to provide a compact view of the robot behavior in the form of a FSA. This attractive idea stems from the second pattern observed, concerning the composition of the dynamics. In particular, the dynamics are composed by different clusters of states, each devoted to the execution of a specific set of actions. The transition between two different clusters leads to the execution of different actions.

To analyze the dynamics of a BN controlling a robot, we collected its trajectories by simulating the experiment. Then, we gathered the trajectories and we generated a graph of the observed state transitions. We performed such procedure on the best performing networks of both the experiments. For lack of space, the graphs can be found as on-line supplementary material [9].

The state space of the robot performing corridor navigation can be decomposed in three macro areas. One is responsible of the behaviors that react to walls detected on the east side of the robot. Likewise, another cluster of states is devoted to avoid the obstacles on the west side of the robot. Besides, the two areas are both connected to a third cluster, responsible of moving the robot straight ahead as long as no obstacle is detected. This analysis reveals that the constant reactivity required to solve this task and the compactness of the dynamics are achieved by continuously reusing the same areas to perform the right action. The continuous sensing of the environment guides the transitions between the clusters of states, allowing the robot to be constantly responsive.

Analyzing the state space of the sequence recognition network, the two patterns are immediately noticeable: the very limited number of states and the compelling order. Again, we observe the organization in sets of nodes devoted to the execution of different actions. The memory is implicitly encoded through this organization: at the top of the graph a series of nodes allow the robot to navigate on the background with its LEDs off until the first colored stripe is found. Then, two sets of nodes are responsible of the next action, depending on the detected color (turn LEDs on if black, turn LEDs off if gray). Once the first color has been recognized, the BN goes into a new region, dual to the first. Here, we find another area for the background color and two sets of nodes for the black and gray with actions switched with respect to the first region. When also the second color is recognized, the dynamical behavior returns back to the first area, reusing the same states to recognize a sequence of any length. This analysis shows that the memory, in our case the last seen color, is stored in the state space in which the BN operates.

The observation of the two state spaces, along with the analysis performed so far, suggests the idea: starting from the graph of a BN state space, it is possible to provide a view of its behavior in the form of a FSA. In fact, each cluster of states contains few topical states, visited several times, and a series of other nodes gradually increasing in number and decreasing in visits. To verify this property, we performed the analysis of the graph for all the final networks of the corridor navigation test case. We reported on a log-log plot the distribution of the states against the number of visits. The results, showed in Figure 7 for two typical cases, suggest that the dynamical behavior of a BN is built around few, prominent states that correspond to the main traits of the robot behavior. Exploiting such property, we followed a simple heuristics to provide, starting from state space graphs of some typical successful networks, the corresponding FSA representations. Following the heuristics, we choose the automaton states by starting the observation from the topical states and gradually moving to the less important ones. The result is that a state in the FSA takes the place of a clusters of connected states in the phase space in which the BN remains until a specific input is received. We report the FSA derived from the sequence recognition graph in Figure 8.

Summarizing, we can say that this is a first heuristics which exploits the automatic design of BN-robot systems to synthesize FSA description of behaviors able to perform the desired tasks. This is just a first proof of concept, further analysis and a formal definition are required. The next step will be to join the two test cases and carry out the same analysis described. In this manner, the two characterizing aspects, dynamic nature and memory, can be studied

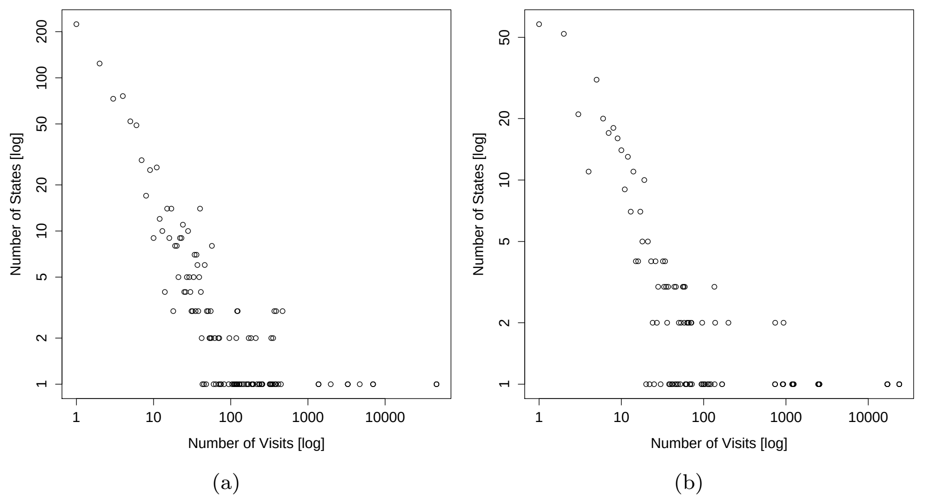


Figure 7: Distribution of the states against the number of visits in two typical cases

together in a complex environment.

The perspective of the work might be to become a good alternative to other methods for the automatic design of FSA, most of which are based on evolutionary programming [1, 22, 7]. More precisely, our approach leads to some advantages with respect to the open issues in EP [14]: we obtain the FSA as an indirect product of the design process of BN-robot systems and, hence, we do not have the problems of representing automata in terms of genomes and constraints. Moreover, in EP it is normally required to impose constraints to contain the dimensions of the automata while, as discussed, our approach has intrinsically the property of producing minimal results. However, in-depth analysis must be carried out, by performing a full quantitative comparison of the two approaches.

5 Conclusion

In this paper, we have exploited the properties of the automatic design of BN-robot control systems to synthesize FSA representations of the controllers. This result has been obtained through a series of analysis performed on the state space of the successful networks. In particular, the exploration revealed two crucial patterns: the minimality of the results and their strong internal organization. The former aspect lies in the extremely tiny portion of state space traversed by the dynamics of the successful networks. The latter concerns the organization of these dynamics in clusters of states occupying different areas of the state space, each corresponding to a different set of actions to perform.

These results allowed us to outline a simple heuristic to derive a compact view of the best performing network behaviors in terms of FSA. Future work concerns the definition of a formal procedure, possibly an automatic method, and its detailed analysis. Finally, a complete quantitative comparison with the

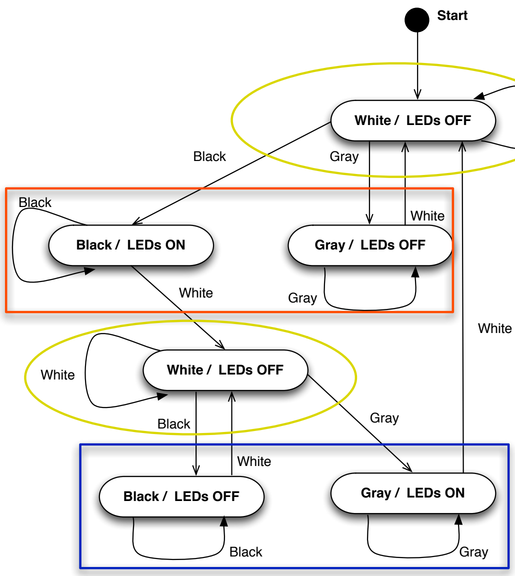


Figure 8: Finite state automaton representation of the state space graph. Sequence learning test case

existing methods for the automatic design of FSA, e.g. evolutionary programming, will give deep insight into the perspectives of the proposed work.

References

[1] P. Angeline and J. Pollack. Evolutionary module acquisition. In *Proceedings of the Second Annual Conference on Evolutionary Programming*, pages 154–163. MIT Press, Cambridge, MA, 1993.

[2] L. Ansaloni, M. Villani, and R. Serra. Dynamical critical systems for information processing: a preliminary study. In M. Villani and S. Cagnoni, editors, *Proceedings of the Satellite Workshops of the International Conference of the Italian Association for Artificial Intelligence (AIIA09)*, pages 210–218. Reggio-Emilia, Italy, 2009. Published on CD.

[3] Y. Bar-Yam. *Dynamics of complex systems. Studies in nonlinearity.* Addison-Wesley, Reading, MA, 1997.

[4] M. Dorigo. Learning by probabilistic boolean networks. In M. Wada D. Ruck and editors D. Bounds, editors, *1994 IEEE International Conference on Neural Networks: IEEE World Congress on Computational Intelligence*, pages 887–891. IEEE Press, Piscataway, NJ, 1994.

[5] D. B. Fogel. *Evolving artificial intelligence.* PhD thesis, La Jolla, CA, USA, 1992. UMI Order No. GAX93-03240.

[6] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution.* John Wiley, New York, USA, 1966.

[7] L.J. Fogel, P.J. Angeline, and D.B. Fogel. An evolutionary programming approach to self-adaptation on finite state machines. In *Proceedings of the fourth annual conference on evolutionary programming*, pages 355–365. MIT Press, Cambridge, MA, 1995.

[8] C. Fretter and B. Drossel. Response of boolean networks to perturbations. *The European Physical Journal B - Condensed Matter and Complex Systems*, 62(3):365–371, 2008.

[9] L. Garattoni, C. Pinciroli, A. Roli, M. Amaducci, and M. Birattari. Additional material to the paper "finite state automata synthesis in boolean network robotics" available at. `http://iridia.ulb.ac.be/supp/IridiaSupp2012-014/`.

[10] C. Gershenson. Introduction to random boolean networks. In *Workshop and Tutorial Proceedings, Ninth International Conference on the Simulation and Synthesis of Living Systems (ALife IX)*, pages 160–173. MIT Press, Cambridge, MA, 2004.

[11] S.A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437–467, 1969.

[12] S.A. Kauffman. Antichaos and Adaptation. *Scientific American*, 265:78–84, 1991.

[13] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.C Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering. In Paulo J.S. GonÃ§alves, Paulo J.D. Torres, and Carlos M.O. Alves, editors, *Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, pages 59–65. IPCB, Castelo Branco, Portugal, 2009.

[14] M. O'Neill, L. Vanneschi, S. Gustafson, and W. Banzhaf. Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, 11(3-4):339–363, 2010.

[15] S. Patarnello and P. Carnevali. Learning networks of neurons with boolean logic. *Europhysics Letters*, 4(4):503–508, 1986.

[16] P. Petrovic. Strengths and weaknesses of fsa representation. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 723–725. ACM, New York, NY, USA, 2007.

[17] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, T. Stirling, Á. Gutiérrez, L. M. Gambardella, and M. Dorigo. ARGoS: a modular, multi-engine simulator for heterogeneous swarm robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011)*, pages 5027–5034. IEEE Press, Los Alamitos, CA, 2011.

[18] A.S. Ribeiro, S.A. Kauffman, J. Lloyd-Price, B. Samuelsson, and J.E.S. Socolar. Mutual information in random boolean models of regulatory networks. *Physical Review E*, 77(1):011901, 2008.

[19] A. Roli, M. Manfroni, C. Pinciroli, and M. Birattari. On the design of boolean network robots. In *Proceedings of EVOApplications 2011*, Lecture Notes in Computer Science, pages 43–52. Springer, Berlin, Germany, 2011.

[20] R. Serra, M. Villani, A. Graudenzi, and S.A. Kauffman. Why a simple model of genetic regulatory networks describes the distribution of avalanches in gene expression data. *Journal of Theoretical Biology*, 246(3):449–460, 2007.

[21] R. Serra and G. Zanarini. *Complex Systems and Cognitive Processes*. Springer-Verlag, Secaucus, NJ, 1990.

[22] W. Spears and D. Gordon. Evolving finite-state machine strategies for protecting resources. In *Foundations of Intelligent Systems*, volume 1932 of *Lecture Notes in Computer Science*, pages 5–28. Springer, Berlin , Germany, 2010.

[23] R. Sun and C.L. Giles. Sequence Learning: From Recognition and Prediction to Sequential Decision Making. *IEEE Intelligent Systems*, 16(4):67–70, 2001.